UNIVERSIDADE
DE VIGO

ESCOLA SUPERIOR DE ENXEÑARÍA INFORMÁTICA

Memoria do Traballo de Fin de Grao que presenta

D. Julián Jiménez González

para a obtención do Título de Graduado en Enxeñaría Informática

SuperCars: Desarrollo de videojuego para dispositivos móviles

Maio, 2015

Traballo de Fin de Grao Nº: EI 14/15 - 010

Titor/a: Alma María Gómez Rodríguez
Co-titor/a: David Ramos Valcárcel
Área de coñecemento: Linguaxes e Sistemas Informáticos
Departamento: Informática

Índice

1. DESCRIPCIÓN DEL PROYECTO 5

1.1. INTRODUCCIÓN 5
1.2. OBJETIVOS 6
1.3. DOCUMENTACIÓN ENTREGADA 7
1.3.1. Memoria de Trabajo 7
1.3.2. Manual de Usuario 7
1.3.3. Manual Técnico 7
1.3.4. Código fuente 8

2. PROCESO DE DESARROLLO, TECNOLOGÍAS Y HERRAMIENTAS 8

2.1. PROCESO DE DESARROLLO 8
2.2. TECNOLOGÍAS Y HERRAMIENTAS 9
2.2.1. Cocos2d-x v3.2 (C++) 9
2.2.2. Android NDK v.r9d 9
2.2.3. Python v2.7.8 10
2.2.4. Apache Ant v1.9.4 10
2.2.5. Eclipse ADT Bundle 10
2.2.6. Adobe Photoshop CS6 10
2.2.7. Tiled v0.11 10
2.2.8. Visual Paradigm v11.2 11
2.2.9. Harvestapp 11

3. PLANIFICACIÓN Y COSTES 11

3.1. PLANIFICACIÓN TEMPORAL 11
3.2. SEGUIMIENTO 12
3.2.1. Reparto temporal real 13
3.2.2. Funcionalidades por Sprint 13
3.2.2.1 Sprint 1. Implementación de los controles del vehículo 13
3.2.2.2 Sprint 2. Adición de circuito y colisiones 13
3.2.2.3 Sprint 3. Creación de menús 14
3.2.2.4 Sprint 4. Incorporación de contrincantes 14
3.2.2.5 Sprint 5. Modificaciones de vehículo 15
3.2.2.6 Sprint 6. Mejoras y pruebas globales 15
3.2.3. Desvíos 15
3.2.3.1 Cambio en el orden de *Sprints* 15
3.2.3.2 Implementación de Box2D 15
3.3. ESTIMACIÓN DE COSTES 15
3.3.1. Costes de Personal 16
3.3.2. Costes Software 16
3.3.3. Costes Hardware 16

4. PROBLEMAS, SOLUCIONES Y CONCLUSIONES 17

4.1. PROBLEMAS ENCONTRADOS 17
4.1.1. Implementación de Box2D 17
4.1.2. Elevado coste de horas de optimización 18
4.1.3. Dificultad para documentar 18
4.1.4. Incompatibilidad del tipo String de JSON 18
4.2. CONCLUSIONES 19
4.2.1. Tecnologías y herramientas 19
4.2.2. Planificación y gestión 19
4.2.3. Personal 19

5. MEJORAS Y POSIBLES AMPLIACIONES 20

5.1. Portabilidad a otras plataformas 20
5.2. Mejorar las mecánicas del juego 20
5.3. Mejorar el algoritmo de generación de obstáculos 20
5.4. Mejorar los efectos especiales 21
5.5. Mejorar la IA de los rivales 21
6. REFERENCIAS Y BIBLIOGRAFÍA 21
6.1. Artículos 21
6.2. Libros 22
6.3. Documentaciones técnicas 22
6.4. Otros 22

1. Descripción del Proyecto

1.1. Introducción

Desde que se comenzaron a comercializar en la década de los 90, el concepto de "videojuego" no ha hecho más que evolucionar hasta lo que conocemos hoy en día, llegando incluso en algunos países al estatus de medio artístico. En EEUU, el MoMA[1] ya ha introducido un buen número de videojuegos entre sus colecciones y exposiciones (P. Antonelli, 2012). Tanto es así que, según el informe anual de 2014 de NewZoo (*Ilustración 1*), la industria del videojuego generó más de $80 billones y cuenta con más de 1700 millones de jugadores en todo el Mundo.

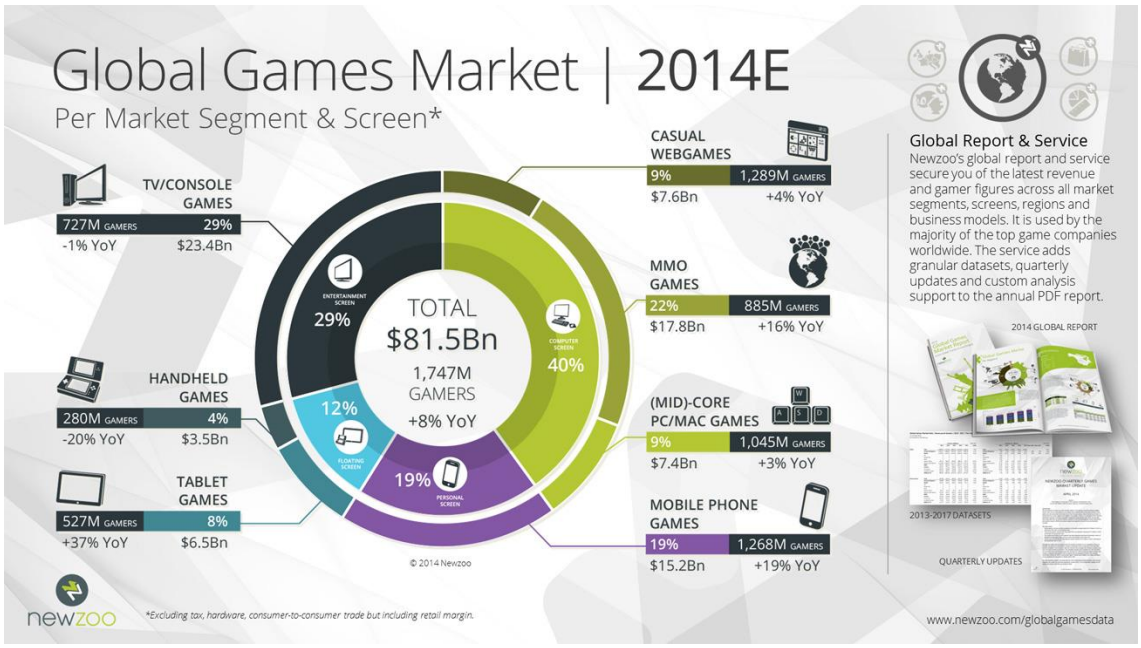


Ilustración 1. *Informe segmentado de NewZoo sobre la industria del videojuego en 2014*

Se considera que, en 2015, se podrían alcanzar los $100 billones, poniéndose a la altura de otras industrias de entretenimiento como la música y el cine. Esta situación favorece el aumento de opciones profesionales en el sector, ya que cada vez más disciplinas se ven involucradas en el proceso de desarrollo de un videojuego. Entre otras, podemos encontrar:

- **Desarrollo de Software y herramientas.** Programadores junior/senior, ingenieros de Software, físicos, matemáticos…
- **Sonido.** Ingenieros de sonido, músicos, compositores, cantantes…
- **Arte.** Modeladores 3D, pintores, dibujantes, diseñadores, artistas gráficos, escritores...
- **Telecomunicaciones.** Ingenieros de Telecomunicaciones, ingenieros en Tecnologías de la Información…

[1] *Museum of Modern Art (Nueva York)*

Por otro lado, existe mucha documentación y herramientas que permiten a cualquier persona desarrollar su propio videojuego, tanto si ya tiene los conocimientos necesarios como si está interesada en aprenderlos. También hay muchas plataformas de ayuda mutua para desarrolladores de Software en general (como *StackExchange* o *reddit*), además de foros específicos y soporte de los propios desarrolladores de cada herramienta.

Entre las distintas plataformas de videojuegos, la de dispositivos móviles (*Smartphones* y *Tablets*) destaca como la de mayor auge en la actualidad, y (como refleja la *Ilustración 1*) la que mayor crecimiento anual prevé. Por su parte, el SO Android cuenta con una cuota del mercado global de dispositivos móviles de alrededor del 85% (*Ilustración 2*), siendo videojuegos más del 40% de las aplicaciones descargadas para estos dispositivos.

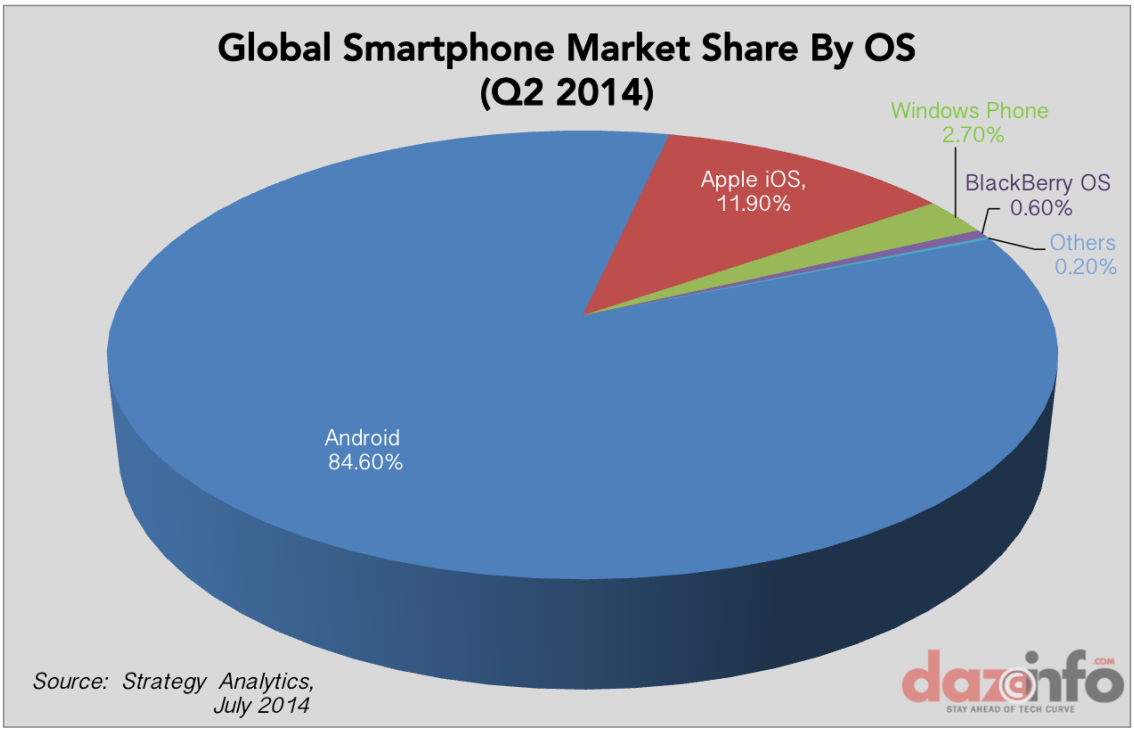


Ilustración 2. *Distribución global de SO para Smartphones 2014 (Dazeinfo)*

Esto puede relacionarse directamente con el creciente interés de las personas por los juegos "casual", a los que no necesitan dedicar mucho tiempo y que les permiten distraerse durante unos minutos de sus actividades cotidianas. Esto, unido al reducido coste que tienen, los hace mucho más accesibles para todo el mundo.

Por todo ello, se ha decidido realizar como TFG[2] un videojuego para Android.

1.2. Objetivos

El objetivo de este proyecto es desarrollar un videojuego de carreras en 2D de un jugador para Android.

[2] *Trabajo de Fin de Grado*

Desde los menús se podrán realizar las configuraciones mínimas de la carrera y competidores (número de vueltas, dificultad, número de contrincantes…).

Durante la carrera se tendrá una vista aérea y se podrán ver los contadores de tiempo, vueltas y posición en carrera. Los rivales tendrán distintos niveles de dificultad. Esto influirá en los parámetros de su vehículo o su comportamiento durante la carrera.

Además de la línea del circuito, habrá zonas que modifiquen el comportamiento del vehículo (como arena o hierba) y muros que lo limiten.

También se guardará un ranking de vueltas rápidas.

1.3. Documentación entregada

La documentación correspondiente a este TFG es la siguiente:

1.3.1. Memoria de Trabajo

Este mismo documento, en el cual se recoge toda la información referente a la planificación y gestión del proyecto, así como los desvíos producidos en tiempo y forma con respecto a dicha planificación.

1.3.2. Manual de Usuario

Documento en el que se describe de forma visual cómo funciona la aplicación de cara a los usuarios finales de la misma.

1.3.3. Manual Técnico

En él, se detalla la aplicación desde el punto de vista de un desarrollador, facilitando las tareas de mantenimiento y ampliación de la misma.

Nótese que toda la documentación de diseño de la aplicación se presenta sólo en su versión final, a pesar de que se planteó al inicio del proyecto que se detallaran las distintas fases por las que ha pasado la documentación. Esto se debe a que se considera que buena parte de la información sería redundante, ya que los cambios entre una fase y otra no tienen por qué ser sustanciales.

1.3.4. Código fuente

En un CD/DVD, se adjuntarán el código fuente y la documentación descrita correspondientes a la aplicación, con la estructura siguiente:

- **/codigo**. Directorio que contendrá los archivos del código fuente.
- **/doc**. Donde se incluirá toda la documentación.
- **SuperCars.apk**. Archivo de instalación de la aplicación. Para instalar la aplicación desde este archivo es necesario permitir en el dispositivo Android la instalación de aplicaciones de orígenes desconocidos[3].

2. Proceso de desarrollo, tecnologías y herramientas

Teniendo en cuenta la inexperiencia del desarrollador con el entorno de desarrollo y las tecnologías utilizadas para este proyecto, se ha considerado que un desarrollo ágil (inspirado en *SCRUM* y dirigido por *Sprints*) es lo más adecuado, permitiendo adaptarse a posibles cambios o problemas que surjan durante el proyecto.

En ningún caso se hace referencia a *SCRUM* como proceso de desarrollo del proyecto, sino como una fuente de inspiración para utilizar un proceso propio. Esto se debe, entre otras cosas, a que:

- *SCRUM* es un modelo de referencia para equipos de desarrollo.
- No tiene sentido unir los roles mínimos (*Scrum Master*, *Product Owner* y *Development Team*) en una única persona.
- No tiene sentido realizar *Daily Meetings*[4] siendo un único desarrollador.
- No existe un cliente con el que revisar el trabajo completado tras cada Sprint.

2.1. Proceso de desarrollo

Se ha realizado, como se indica anteriormente, un desarrollo basado en *Sprints* que añaden funcionalidad de forma iterativa e incremental a la aplicación. En todos los *Sprint* se realizan diversas tareas en mayor o menor medida, siendo éstas:

- **Análisis del alcance y la funcionalidad.** En todos los *Sprint* se ha decidido, al inicio del mismo, cuáles serían las funcionalidades principales a desarrollar.
- **Implementación.** Todas las fases del desarrollo han tenido una parte importante de implementación de funcionalidades.
- **Pruebas.** En todos los *Sprint* se han realizado pruebas para verificar el correcto funcionamiento de las funcionalidades implementadas. Conforme el proyecto ha avanzado, estas pruebas han tenido un mayor peso, ya que hay más interacción entre distintas partes del sistema.
- **Documentación.** Cada funcionalidad conlleva la creación de nueva documentación que la apoye o la modificación de la que ya existía en fases previas.

3 Cualquier aplicación que no se instale directamente desde Google Play Store es considerada de orígenes desconocidos.
4 Reuniones que realizan los equipos de desarrollo basados en SCRUM para compartir el estado actual de su trabajo, si han tenido problemas para hacerlo y qué planean hacer.

La división prevista inicialmente en *Sprints* es la siguiente:

- Sprint 1. Implementación de los controles del vehículo.
- Sprint 2. Adición de circuito y colisiones.
- Sprint 3. Incorporación de contrincantes.
- Sprint 4. Creación de menús.
- Sprint 5. Modificaciones de vehículo.
- Sprint 6. Mejoras y pruebas globales.

Esta división está sujeta a cambios debidamente justificados, pudiéndose además modificar una funcionalidad implementada anteriormente si se encuentra algún error, o si esto es necesario para desarrollar el propio *Sprint*.

Todos los detalles sobre la distribución real de los *Sprints* se pueden encontrar en el apartado *3.2.2*.

2.2. Tecnologías y herramientas

A continuación, se listan las tecnologías y herramientas empleadas, así como su utilidad para el desarrollo del proyecto.

2.2.1. Cocos2d-x v3.2 (C++)

Tras estudiar otras posibilidades como *Unity3D*, *AndEngine* o *LibGDX*, éste ha sido el *framework* seleccionado para desarrollar el proyecto. Las razones principales son dos:

- Es una herramienta open-source y gratuita, tanto para proyectos comerciales como no comerciales.
- Posee una documentación bastante completa para iniciar el desarrollo, así como una comunidad muy activa y amplia.

Además, *Cocos2d-x* emplea C++ como lenguaje de programación, siendo el más extendido en el desarrollo profesional de videojuegos y que ofrece un gran rendimiento en aplicaciones que precisen cargas de trabajo intensas de la CPU.

Por otra parte, *Cocos2d-x* tiene una capa muy reducida dependiente de la plataforma, por lo que permite exportar con pocas variaciones el proyecto a distintas plataformas (iOS, Android, Linux, Win32…).

2.2.2. Android NDK v.r9d

Android NDK es una herramienta que permite implementar partes de una aplicación Android en código nativo (como C++), a cambio de un aumento en la complejidad de dicha aplicación, al provocar que la ejecución de la aplicación sea directamente sobre el procesador y no interpretada por una máquina virtual. Por lo tanto, si bien no es recomendable para cualquier aplicación, sí tiene cabida en aplicaciones que, como se ha comentado anteriormente, requieren una carga de trabajo importante de la CPU.

Además, es necesaria alguna herramienta que permita compilar código C++ en Android.

NOTA: En el momento en que se inicia el desarrollo, la versión r9d es la óptima para trabajar con *Cocos2d-x*, no siendo válidas versiones superiores.

2.2.3. Python v2.7.8

Python es un lenguaje de programación interpretado de alto nivel. Se considera un lenguaje muy sencillo que enfatiza en la legibilidad del código y el dinamismo, siendo muy utilizado para *RAD*[5] y como conector entre varios componentes o módulos escritos en C++ (*'Comparing Python to other languages'*). Es por esto que *Cocos2d-x* v3.0+ lo emplea para configurar y compilar los archivos de la aplicación.

NOTA: debido a ciertos cambios en la sintaxis de Python, *Cocos2d-x* requiere que se utilice Python v2.

2.2.4. Apache Ant v1.9.4

Apache Ant es una herramienta que se utiliza para la automatización de procesos de compilación y construcción. Su función es similar a la de *Make*, pero no depende de las órdenes del *Shell* de cada *SO*[6] (gracias a la *Java Virtual Machine*) y es más eficaz a la hora de determinar qué archivos están actualizados.

Es, por lo tanto, una herramienta ideal para soluciones multi-plataforma, al igual que *Cocos2d-x*.

2.2.5. Eclipse ADT Bundle

Éste paquete contiene el *IDE*[7] Eclipse junto a Android *SDK*[8] (ya configurado), por lo que es una opción muy recomendable para desarrollar, aunque no imprescindible.

2.2.6. Adobe Photoshop CS6

Photoshop es un programa de edición de imagen 2D necesario en un proyecto como éste, siendo utilizado para múltiples tareas como recortar, escalar y orientar las imágenes que componen el juego.

2.2.7. Tiled v0.11

Tiled es una herramienta que permite crear escenarios para juegos 2D con orientación ortogonal, isométrica o hexagonal. Utiliza una imagen que contiene *tiles* ("baldosas") de un tamaño uniforme, para crear un mapa apilando esos *tiles* y que permite configurar puntos de *spawn*[9], colisión…

---

[5] *Rapid Application Development*
[6] *Sistema Operativo*
[7] *Integrated Development Environment*
[8] *Software Development Kit*
[9] *El punto de spawn es como se conoce al lugar donde un personaje de un videojuego aparece por primera vez o reaparece después de morir.*

La ventaja de utilizar un mapa en formato *.tmx* (similar a *XML*), además de lo ya mencionado, es que es muy sencillo abrirlo en el editor y sustituir un *tile* por otro en cualquier momento, para cambiar por ejemplo la disposición de los obstáculos u ornamentos.

2.2.8. Visual Paradigm v11.2

Visual Paradigm es un programa para el diseño de Software que emplea *UML*[10]. Se ha utilizado para crear los diagramas de análisis y diseño del sistema (disponibles en el Manual Técnico).

2.2.9. Harvestapp

Esta es una herramienta online que facilita el seguimiento de horas de un proyecto. Se pueden introducir las horas trabajadas en el día, o iniciar/parar un temporizador en tiempo real. Esto permite llevar un control exhaustivo de las horas empleadas, además de poder especificar tareas (en este caso se ha dividido por *Sprint*) o incluso precio.

3. Planificación y costes

En este apartado se detallan la planificación temporal prevista (*3.1*) y la adecuación real del proyecto a ésta (*3.2*). También se describe qué funcionalidades se han desarrollado en cada *Sprint* (*3.2.2*) y la justificación correspondiente a los desvíos producidos (*3.2.3*).

3.1. Planificación temporal

La planificación realizada inicialmente para la distribución de horas, teniendo en cuenta que se considera que el proyecto debe abarcar un total de 300 horas, es la siguiente:

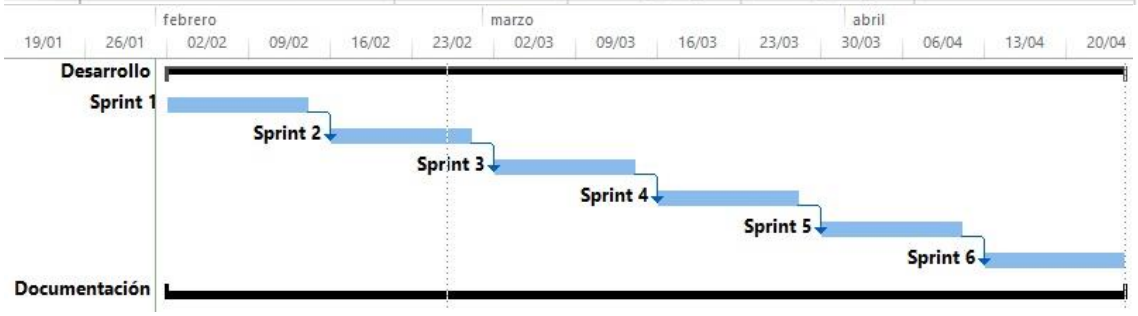
Dedicación semanal prevista (horas)	25

Fase	Estimación temporal (semanas)
Sprint 1	2
Sprint 2	2
Sprint 3	2
Sprint 4	2
Sprint 5	2
Sprint 6	2
Total Proyecto	12

A continuación se muestra el Diagrama de Gantt que representa la disposición de horas a lo largo del tiempo de desarrollo estimado.

[10] *Unified Modeling Language*

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Desarrollo**	**12 sem.**	**lun 02/02/15**	**vie 24/04/15**	
2	Sprint 1	2 sem.	lun 02/02/15	vie 13/02/15	
3	Sprint 2	2 sem.	lun 16/02/15	vie 27/02/15	2
4	Sprint 3	2 sem.	lun 02/03/15	vie 13/03/15	3
5	Sprint 4	2 sem.	lun 16/03/15	vie 27/03/15	4
6	Sprint 5	2 sem.	lun 30/03/15	vie 10/04/15	5
7	Sprint 6	2 sem.	lun 13/04/15	vie 24/04/15	6
8	Documentación	12 sem.	lun 02/02/15	vie 24/04/15	



Este método de desarrollo es susceptible a cambios durante el mismo.

3.2. Seguimiento

A pesar de la planificación, no ha sido posible mantener un ritmo constante de trabajo durante todo el periodo de desarrollo, por lo que se han producido desvíos en los plazos de finalización de cada sprint.

A continuación, se presenta el reparto real de horas del proyecto. Posteriormente, se detallarán ciertos desvíos (*3.2.3*) causados por los problemas encontrados durante el desarrollo.

3.2.1. Reparto temporal real

Aquí se detalla en forma de tabla cuál ha sido el reparto de horas entre los distintos *Sprint*, así como la suma total de las horas reales que han sido necesarias para finalizar en tiempo y forma este proyecto.
Se ha decidido no crear un diagrama de Gantt (sí se hizo en la planificación) debido al reparto tan irregular de horas durante todo el proyecto.

Fase	Estimación temporal (semanas)
Sprint 1	87,27
Sprint 2	42,65
Sprint 3	44,52
Sprint 4	48,44
Sprint 5	47,57
Sprint 6	9,64
Total	**280,09**

3.2.2. Funcionalidades por Sprint

A continuación, se detallan las funcionalidades desarrolladas en cada *Sprint*. Éstas han sido decididas al inicio de los mismos, en función de:

- Funcionalidades ya implementadas.
- Funcionalidades restantes.
- Tiempo de desarrollo real con respecto al estimado.

En la descripción de las funcionalidades desarrolladas en cada *Sprint*, no se tienen en cuenta las tareas de documentación, ya que ésta va asociada en mayor o menor medida a cada una de las funcionalidades.

Nótese que se realizó *"Sprint 3. Creación de menús"* antes que *"Sprint 4. Incorporación de contrincantes"*, a pesar de que inicialmente estaba planeado que estos *Sprints* se realizaran en orden inverso (más detalles en *3.2.3.1*).

3.2.2.1 Sprint 1. Implementación de los controles del vehículo

En este *Sprint* se implementa una primera versión de la escena de carrera, con un vehículo que se puede manejar a izquierda y derecha con dos flechas.

- Escena principal de carrera.
- Objeto jugador.
- Botones para controlar al jugador y eventos asociados a éstos.
- Mapa simple (estático).

3.2.2.2 Sprint 2. Adición de circuito y colisiones

En este *Sprint*, se cambia el objeto mapa de una imagen de fondo estática a un objeto de tipo *TMXTiled*, que se mueve hacia abajo y crea la ilusión de ser "infinito".

Se añaden obstáculos simples que se crean de forma aleatoria y el jugador puede colisionar con ellos.

También se añade música de fondo y un efecto sonoro de colisión, así como las líneas de meta de cada vuelta y un texto que indica las vueltas superadas.

- Mapa *TMXTiled*, dinámico e infinito.
- Creación aleatoria de obstáculos y movimiento de los mismos.
- Control de colisiones entre jugador y obstáculos.
- Música de fondo y efectos sonoros básicos.
- Creación de líneas de vuelta y conteo de las mismas.

3.2.2.3 Sprint 3. Creación de menús

En este *Sprint*, se diseñan y crean los menús del juego (escenas), con los botones/objetos correspondientes y toda la navegación entre escenas que se precisa para la versión final del juego. Esto incluye el envío de información entre escenas, como pueden ser los parámetros de la carrera introducidos en el menú de configuración.

También se diseñan e introducen el logotipo del juego y el icono de la aplicación.

- Creación de los menús del juego (y botones correspondientes):
  - Menú principal.
  - Menú de configuración de carrera.
  - Menú de pausa durante la carrera.
  - Menú de fin de carrera.
- Adición de GUI (Graphic User Interface) durante la carrera.

3.2.2.4 Sprint 4. Incorporación de contrincantes

Este *Sprint* se centra en la adición de rivales a la carrera, así como crear un temporizador para guardar un registro del tiempo de cada vuelta. Esto permitirá crear un *ranking* de vueltas rápidas asociadas al nombre introducido por el jugador en la escena de configuración de la carrera.

También se mejora el algoritmo de creación de obstáculos.

- Creación de rivales.
- Creación de temporizador de vueltas.
- Mejora de creación de obstáculos.
- Creación de *ranking* de vueltas rápidas de carrera y total.
- Modificación del mapa para que se vea mejor.

3.2.2.5 Sprint 5. Modificaciones de vehículo

En este Sprint se han desarrollado las funcionalidades finales y se han pulido ciertas características

- Adición de selección de vehículo.
- Modificación de lógica de rivales para evitar obstáculos.
- Mejora de *ranking* y reseteo del mismo.

3.2.2.6 Sprint 6. Mejoras y pruebas globales

Este *Sprint* se ha destinado a solucionar posibles problemas encontrados durante las pruebas finales, así como finalizar y ajustar toda la documentación.

3.2.3. Desvíos

En este apartado se detallan y justifican los desvíos producidos con respecto a la planificación inicial.

3.2.3.1 Cambio en el orden de *Sprints*

Durante el transcurso del proyecto, se decidió realizar *"Sprint 3. Creación de menús"* antes que *"Sprint 4. Incorporación de contrincantes"*. Antes de iniciar este *Sprint*, se había desarrollado la mayor parte de la escena de carrera, y se consideró que sería interesante poder empezar a configurar la misma sin tener que modificar y recompilar el código fuente. Además, la creación de los menús permitiría probar varias carreras sin la necesidad de cerrar la aplicación y abrirla de nuevo.

Éste cambio no implicó ningún desvío en el reparto de horas, ya que a pesar de la variación en el orden, ambos *Sprint* se desarrollaron.

3.2.3.2 Implementación de Box2D

El problema de *Implementación de Box2D* implicó cambios en el código, así como muchas horas de búsqueda de información y ayuda, lo que provocó un desvío aproximado de:

Tarea	Desvío (horas)
Investigación	20
Desarrollo	10
Búsqueda de razones y soluciones correspondientes	20
Implementación de solución final	10
Total	**60,00**

3.3. Estimación de costes

Los costes estimados para el proyecto se detallan a continuación. Se diferencia entre costes de personal, Software y Hardware.

La estimación de costes se realiza teniendo en cuenta la estimación temporal realizada anteriormente (*3.1*). Esto influye directamente en el coste de las licencias Software, que se pagan por mes. En cuanto a los costes de Hardware, se estima durante cuánto tiempo pueden ser útiles los componentes (aún fuera del contexto de este proyecto), para aplicar el porcentaje correspondiente a los 3 meses de desarrollo.

Recurso	Coste
Personal	6.000,00 €
Software	119,52 €
Hardware	49,00 €
	6.168,52 €

3.3.1. Costes de Personal

Los costes de personal están calculados teniendo en cuenta el tiempo de desarrollo (en horas) estimado para un solo desarrollador.

Personal	Horas de trabajo	€/hora	Total
Ingeniero de Software	300	20	6.000,00 €
			6.000,00 €

3.3.2. Costes Software

Los costes de Software se calculan al sumar el precio de todas las licencias de Software de terceros requeridas para el desarrollo del proyecto. El precio de estas licencias puede variar durante el desarrollo y después del mismo, por lo que la estimación refleja el precio de las licencias en el momento de redactar este documento.

Licencia	Precio(/mes)	Total
Adobe Creative Cloud	19,66 €	58,98 €
Office 365	7,00 €	21,00 €
Visual Paradigm Standard	13,21 €	39,63 €
		119,61 €

3.3.3. Costes Hardware

En este apartado se incluye el coste de los componentes Hardware necesarios para el desarrollo del proyecto. Los componentes aquí descritos no son los mínimos necesarios. Sin embargo, se ha tenido en cuenta el precio del Hardware en el momento de iniciar el proyecto, y se han escogido componentes de gama media.

Componente	Precio Total	Amortización (años)	Precio Final
PC	680,00 €	5	34,00 €
Tablet Android 4.2.2	120,00 €	2	15,00 €
			49,00 €

4. Problemas, soluciones y conclusiones

A continuación se detallan los problemas y dificultades surgidos durante el desarrollo del proyecto, así como las soluciones adoptadas (de ser el caso).

4.1. Problemas encontrados

4.1.1. Implementación de Box2D

Inicialmente se decidió tratar de implementar el vehículo utilizando el motor de físicas *Box2D* (utilizado por *Cocos2d-x*), ya que parecía ser a priori la solución idónea. No se consiguió que dicho motor de físicas funcionase en el proyecto, por lo que hubo que tomar una alternativa.

A pesar de buscar una razón que lo justificase, no se ha podido confirmar ninguna de las posibles razones como causante del problema. Las posibilidades que se barajaron tras muchas horas de investigación son:

- **Tamaño de las imágenes mayor del soportado por el dispositivo.** Se emplearon las mismas imágenes en las versiones con y sin *Box2D*, por lo tanto:
  - **Descartado**

- **Errores en el código fuente.** Se encontró una versión de ejemplo similar a lo esperado en un repositorio público de la plataforma *GitHub*. Se probó el código fuente público y posteriormente, ya que no funcionaba, se contactó con el propietario del repositorio. Éste envió por email otra versión que sí debería funcionar, pero el resultado fue el mismo. Por lo tanto:
  - **Descartado**

- **Incompatibilidad del Hardware disponible.** Es posible que el recurso Hardware empleado para desarrollar la aplicación no soporte la inclusión de Box2D. Por ello se probó en otro modelo, en el que tampoco se podía arrancar la aplicación. Por lo tanto:
  - **No demostrable**

Finalmente, se decidió que el desvío comenzaba a ser elevado y habría que buscar una solución alternativa.

Solución: Simplificar las mecánicas del juego y descartar *Box2D* para el desarrollo.

4.1.2. Elevado coste de horas de optimización

Debido a la naturaleza jugable del proyecto, es necesario que el código sea lo más eficiente posible, para permitir que el número de *fps*[11] se mantenga constante. El hecho de que haya muchas imágenes de forma simultánea en la pantalla, además de otras que no se muestran (como por ejemplo cuando un rival desaparece de la pantalla pero sigue moviéndose), ha dificultado esta tarea, obligando a buscar siempre la mejor solución y no la más rápida o sencilla. A la hora de buscar esta mejor solución, también se ha tenido en cuenta que el proyecto sería ampliable incluso después de finalizado, lo que provoca que algunas funcionalidades hayan requerido un tiempo de desarrollo más elevado del que se podría esperar.

4.1.3. Dificultad para documentar

Este proyecto contiene poca interacción entre usuario y sistema, por lo que la documentación más habitual es insuficiente para ofrecer una vista muy detallada de todo el sistema. Esto implica que la documentación debe reflejar las tareas internas más importantes y cómo se llevan a cabo, para lo que ha hecho falta buscar mucha información y ejemplos acerca del diseño de Software para videojuegos. La mayor parte de la información al respecto no es gratuita y esto ha supuesto un gran inconveniente.

4.1.4. Incompatibilidad del tipo *String* de JSON

JSON es un formato para el intercambio de datos que nació como alternativa a *XML*, y al inicio del proyecto se pretendía emplear dicho formato para generar, guardar y recuperar el *ranking* de vueltas rápidas del juego.

Para enviar información entre escenas, se utiliza la clase propia de *Cocos2d-x UserDefault*, que además guarda esta información en un archivo propio *XML*, facilitando mucho la comunicación dentro del sistema. Al intentar añadir estas vueltas rápidas a un archivo *JSON*, se apreció que había una incompatibilidad entre el tipo de datos *String* que emplea *JSON* y el que utiliza *Cocos2d-x*. Tras alrededor de 8 horas de trabajo, no se consiguió solucionar, por lo que se decidió tomar una solución alternativa.

Es importante destacar que esto no se considera que provocara un desvío en el desarrollo del proyecto, ya que la simplificación de este aspecto del juego compensa el tiempo extra necesario para desarrollarla. Además, 8 horas no se considera una cantidad significativa en un proyecto de 300.

Solución: En lugar de guardar un *ranking* de vuelta rápida de cada jugador, se guarda la vuelta rápida total y se puede reiniciar éste *ranking*.

[11] *Frames per second. Es la cantidad de imágenes por segundo que se muestran en la pantalla para generar sensación de movimiento*

4.2. Conclusiones

A continuación se exponen las conclusiones que se han extraído del proyecto, tanto desde un punto de vista técnico, como de planificación y gestión. También se incluye una valoración personal.

4.2.1. Tecnologías y herramientas

Cocos2d es un *framework* muy útil para que un programador sin excesiva experiencia amplíe sus conocimientos y habilidades, ya que la curva de aprendizaje es suave. Su API es sencilla y está muy bien integrado con otras tecnologías/herramientas que podrían considerarse "estándares" en el ámbito del 2D, a la vez que ofrece funcionalidades muy potentes. Esto lo sitúa entre las mejores opciones para desarrollar juegos o aplicaciones interactivas en 2D. Sin embargo, *Cocos2d* no deja de ser una herramienta profesional y que se utiliza para desarrollos comerciales, debido en gran parte a que posee una comunidad muy activa y participativa.

Por lo tanto, se considera que la decisión de emplear *Cocos2d-x* como *framework* para el desarrollo de este proyecto ha sido acertada, y se volvería a utilizar para otro proyecto de características similares.

4.2.2. Planificación y gestión

El seguimiento del proyecto ha servido para identificar a tiempo los desvíos producidos en tiempo o forma, así como tomar las medidas que se han considerado oportunas y evitar mayores desvíos.

En cuanto a los plazos estimados en la *Planificación temporal (3.1)*, finalmente no han sido cumplidos, así como tampoco se ha cumplido la estimación temporal de cada *Sprint*. Ésta última, en cambio, sí se puede considerar que ha sido relativamente acertada, a pesar de no cumplirse de forma estricta.

En cualquier caso, se han superado los objetivos y el tiempo empleado finalmente ha sido el estimado, aunque su distribución haya variado.

4.2.3. Personal

A nivel personal, este proyecto me ha permitido iniciarme en el desarrollo de videojuegos de la mejor forma posible. Ha sido un proyecto relativamente sencillo pero que me ha permitido comprender y profundizar en los aspectos/conceptos más importantes de un videojuego desde un punto de vista técnico, lo que será de gran ayuda en un futuro. También me ha hecho aplicar los conocimientos teóricos de gestión, planificación y documentación de proyectos que he adquirido a lo largo de mis estudios de Grado, permitiéndome ver de primera mano la utilidad y necesidad de todas éstas actividades en cada proyecto.

Como conclusión personal, éste ha sido un proyecto muy gratificante y que me ha servido para comprobar que desarrollar videojuegos puede disfrutarse tanto como jugar a ellos.

Por otra parte, tras terminar este proyecto creo que los videojuegos (y la programación gráfica en general) son una gran forma de enseñar a la gente a programar, y me hubiera gustado que alguna asignatura de la carrera hubiera implicado algo relacionado o similar.

5. Mejoras y posibles ampliaciones

Debido al tipo de proyecto, cualquier persona que juegue a *SuperCars* puede tener una idea que considere que mejoraría su experiencia jugable. Es por esto que lo mencionado en este apartado no es más que algunas de las muchas posibilidades de ampliación.

5.1. Portabilidad a otras plataformas

Ya que éste proyecto se ha centrado en Android como plataforma de ejecución de la aplicación, es indudable que realizar un *port*[12] a otras plataformas como iOS, Windows o Linux sería una ampliación interesante y que facilitaría la distribución del juego. Sin embargo, ya que éste no es un proyecto comercial, no sería imprescindible portarlo a otra plataforma, más allá de la comodidad o facilidad para algún *stakeholder*[13].

En cualquier caso, haber utilizado Cocos2d-x facilita esta tarea en gran medida, ya que, el propio *framework*, permite compilar las mismas clases C++ y los mismos recursos para las plataformas mencionadas, entre otras. A pesar de ello, habría que comprobar en cada caso que no se produzca ninguna incompatibilidad entre formatos o de otro tipo.

5.2. Mejorar las mecánicas del juego

En este proyecto se ha decidido que establecer 3 posibles posiciones para el jugador en sentido horizontal era una forma de complicar en cierto modo la tarea de esquivar rivales y obstáculos, pero se podría por ejemplo ampliar el número de posiciones a 5. Para esto, habría que ajustar el tamaño de los elementos en pantalla y quizás sería menos cómodo de visualizar, pero no cabe duda de que sería una mejora a tener en cuenta.

5.3. Mejorar el algoritmo de generación de obstáculos

Actualmente, los obstáculos se generan de forma cíclica cada X tiempo, y cambiar esto por otro tipo de generación podría ser interesante. Podría, por ejemplo, darse el caso de que el jugador sólo pueda pasar por el centro ya que, tanto izquierda como derecha, estén bloqueadas. También se podrían variar los tamaños de los obstáculos en función de la dificultad o crear obstáculos que se muevan horizontalmente cruzando la carretera.

---

[12] *Port es como se conoce a la tarea de adaptación de un videojuego diseñado para una plataforma específica a otra plataforma distinta. Es una práctica habitual en la industria.*

[13] *Un stakeholder es todo aquel interesado en el desarrollo de un proyecto en cualquier sentido. Pueden incluír desde inversores, desarrolladores o clientes hasta los gerentes de la empresa, un gobierno o alguna ONG.*

5.4. Mejorar los efectos especiales

Tanto el apartado gráfico como el sonoro son básicos y enormemente mejorables. Añadir otro tipo de sonido para las colisiones o un efecto visual de partículas (como polvo o humo), mejorarían notablemente la experiencia del jugador. También se podrían crear efectos de derrape o cambios climatológicos que dificulten la partida.

5.5. Mejorar la IA de los rivales

Ya que la IA[14] que se ha desarrollado para este proyecto es muy sencilla, es muy susceptible a mejoras. Se podría hacer que, para evitar una colisión, los rivales variasen su velocidad si se encuentran en una situación comprometida (como la descrita en el apartado *5.3*), o que traten de forzar al jugador a colisionar con un obstáculo.

6. Referencias y bibliografía

A continuación se enlazan los documentos o recursos web a los que se hace referencia durante el documento. Nótese que los documentos con enlace web pueden dejar de ser accesibles tras la última fecha de consulta, 21 de mayo de 2015.

6.1. Artículos

P. Antonelli (2012) ‘Video Games: 14 in the Collection, for Starters’, Museum of Modern Art (New York)

Python.org ‘Comparing Python to other languages’

[14] *Inteligencia Artificial*

6.2. Libros

A. Álvarez García, R. de las Heras del Dedo, C. Lasa Gómez (2012) 'Métodos Ágiles y SCRUM'. Anaya.

F. Hussain, A. Gurung, G. Jones, (2014) 'Cocos2d-x Game Development Essentials". Sonar Systems. Packt Publishing.

I. Sommervile, (2005) 'Ingeniería del Software'. 7 Ed. Pearson Addison-Wesley

---

R. Strougo, R. Wenderlich (2012) 'Learning Cocos2D: A Hands-On Guide to Building iOS Games with Cocos2D, Box2D, and Chipmunk'. Pearson Addison-Wesley.

6.3. Documentaciones técnicas

http://www.cplusplus.com/
Web de referencia para desarrolladores de C++.

http://en.cppreference.com/w/
Referencia completa y online de C y C++, así como de librerías estándar de ambos.

http://cocos2d-x.org/wiki/Cocos2d-x
Wiki de Cocos2d-x

http://www.cocos2d-x.org/reference/native-cpp/V3.2/index.html
API Reference de Cocos2d-x v3.2

6.4. Otros

http://discuss.cocos2d-x.org/
Foro oficial de *Cocos2d-x*

http://stackoverflow.com/
Comunidad de *StackExchange* en la que buscar ayuda ante problemas de programación

http://gamedev.stackexchange.com/
Comunidad de *StackExchange* especializada en el desarrollo de videojuegos